Amendments to the Claims

1. (Currently amended) A method of instrumenting a program to provide instrumentation data, the method comprising:

creating an instrumented version of the program comprising duplicate versions of at least some code paths in the programs, such that a duplicate code path has an original version code path and an instrumented version code path with instrumentation code for capturing instrumentation data;

tracking a frequency of execution of the code paths;

when a code path is to be executed, determining to dispatch execution into the instrumented version code path at a sampling rate for the respective code path and otherwise into the original version code path such that, for a given sampling rate, a ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the given sampling rate; and

adapting the sampling rate for the code paths according to the frequency of execution of the code paths, such that, after adapting, a ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the adjusted sampling rate.

2. (Original) The method of claim 1 wherein instrumentation data comprises data relating to runtime data references, branch executions, memory allocations, synchronization events, data loads, data stores, or branches.

3. (Currently amended) A method of instrumenting a program to provide runtime program data, the method comprising:

providing a duplicate version of at least some already present procedures in the program with instrumentation for capturing runtime program data;

executing the duplicate version of at least some of the procedures; and

subsequently, for each of the already present procedures, selectively reducing a frequency at which the duplicate version of the procedure is executed, the frequency equivalent to the number of executions of the duplicate version taken as a percentage of the total number of executions of the procedure.

4. (Original) The method of claim 3 wherein the frequency at which the duplicate version is executed is reduced at a rate inversely proportional to how frequently a procedure of the software is executed.

5. (Original) The method of claim 3 wherein the frequency at which the duplicate version is executed is reduced as a function of how frequently a procedure of the software is executed.

6. (Currently amended) A method of instrumenting a computer program containing procedures, the method comprising:

creating a copy of at least some of the original procedures in the computer program;

inserting instrumentation into the copies;

creating an executable version of the program containing the original procedures and the copies; and

executing the executable version of the program, wherein the copies of the procedures are executed in bursts, and the frequency at which the bursts are performed decreases as the total number of executions of either the original procedure or copy of the procedure is executed, said frequency comprising the number of executions of the copy of the procedure divided by the total number of executions.

7. (Currently amended) A method for detecting memory leaks in software, the method comprising:

creating an instrumented version of the software containing an original version and an instrumented version of each procedure in the software;

executing the instrumented version of the software, wherein the instrumented version of the procedures are sampled at higher rates for procedures whose original versions or copies are executed less frequently and sampled at lower rates for procedures whose original versions or copies are executed more frequently, wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure taken as a percentage of total number of executions of either version of the procedure;

storing instrumentation data obtained by execution of the instrumented version of the software; and

reporting all objects that satisfy a predefined staleness condition as memory leaks.

8. (Original) The method of claim 7 wherein instrumentation data comprises heap allocation, heap free and heap access information.

9. (Original) The method of claim 7 wherein reporting all objects comprises reporting the heap object, responsible allocation, heap frees that deallocated objects created at that allocation site, and the last access to the leaked object.

10. (Original) The method of claim 9 wherein a source code browser highlights the last access to a leaked object.

11. (Original) The method of claim 7 further comprising creating mapping information from the software to facilitate "last access" information.

12. (Previously presented) The method of claim 7 wherein the predefined staleness condition comprises determining whether an object on the heap has not been accessed within a predetermined length of time.

13. (Original) The method of claim 7 wherein the instrumented version of the procedures are sampled at a rate inversely proportional to how frequently a procedure is executed.

14. (Currently amended) A method of analyzing software, the method comprising:

creating an instrumented version of the software containing an original version and an instrumented version of at least some procedures in the software, wherein the instrumented versions comprise instrumentation points;

inserting additional programming code at the instrumentation points that produce runtime information when executed; and

executing the instrumented version of the software, wherein the additional programming code is executed more frequently when located at instrumentation points for procedures that are less frequently executed, and the additional programming code is executed less frequently when located at instrumentation points for procedures that are more frequently executed;

wherein frequency of execution of additional programming code for a given procedure comprises a number of executions of the additional programming code taken as a percentage of total executions of the procedure.

15. (Original) The method of claim 14 wherein runtime information comprises data relating to memory leaks.

16. (Original) The method of claim 14 wherein runtime information comprises data relating to data races.

17. (Original) The method of claim 14 wherein runtime information comprises data relating to invariance.

18. (Currently amended) A method of instrumenting software, the method comprising:

producing a copy of at least some procedures of the software;

inserting instrumentation into the copies; and

sampling a copy of a procedure at a rate inversely proportional to how frequently either the original or the copy of the procedure is executed;

wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure taken as a percentage of total number of executions of either version of the procedure.

19. (Original) The method of claim 18 wherein the instrumentation stores data relating to the software when executed.

20. (Original) The method of claim 19 further comprising providing the stored data to a tool for analysis.

21. (Original) The method of claim 20 wherein the tool detects memory leaks.

22. (Original) The method of claim 20 wherein the tool detects data races.

23. (Currently amended) A method of instrumenting software, the method comprising:

producing a copy of at least some procedures of the software;

inserting instrumentation into the copies; and

sampling a copy of a procedure at higher rates for procedures whose original versions or copies are executed less frequently and sampling a copy of a procedure at lower rates for procedures whose original versions or copies are executed more frequently;

wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure taken as a percentage of total number of executions of either version of the procedure.

24. (Original) The method of claim 23 wherein the instrumentation communicates data relating to the software to a tool.

25. (Original) The method of claim 24 wherein the tool uses the communicated data to analyze the software.